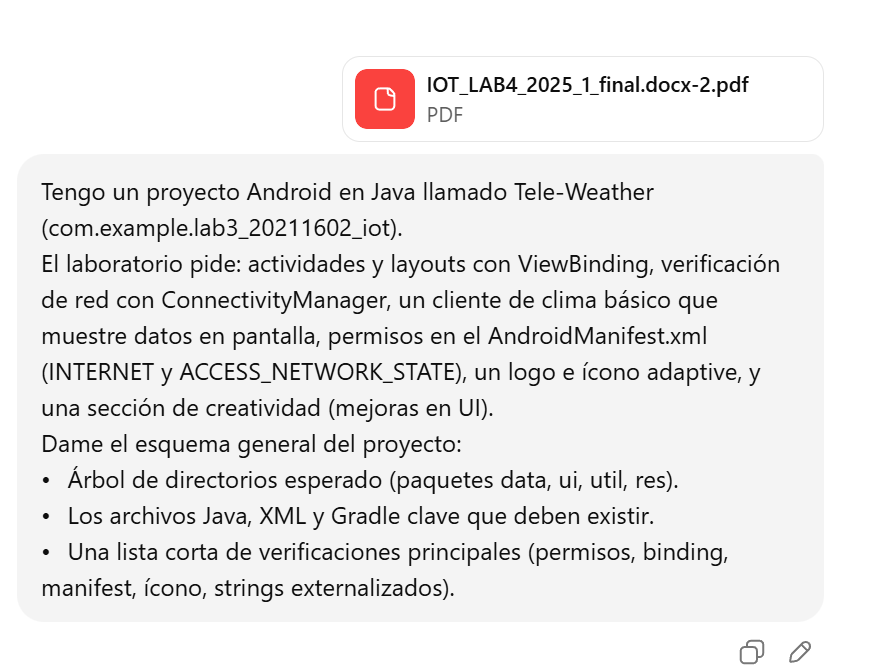
**Uso de Inteligencia Artificial en el desarrollo**

Durante el desarrollo del laboratorio se utilizó la herramienta CHATGPT como apoyo creativo y técnico. A continuación los prompts:

**Estudiante:** Edgar Díaz  
**Código:** 20211602

Prompt 1:



Respuesta:

com.example.lab3\_20211602\_iot

├─ app/ // Capa App (NavHost, temas, arranque)

│ ├─ MainActivity.java // Bienvenida + verificación de red + CTA a Settings

│ ├─ AppActivity.java // Host NavHostFragment + BottomNavigation

│ └─ nav/

│ └─ nav\_graph.xml // Navigation Component (Location / Forecast / Sports)

│

├─ ui/ // Presentación (MVVM)

│ ├─ common/

│ │ ├─ BaseFragment.java // Helpers (ViewBinding, toasts, lifecycle)

│ │ └─ decorations/ // ItemDecorations (espaciados/dividers)

│ ├─ location/

│ │ ├─ LocationFragment.java

│ │ ├─ LocationViewModel.java

│ │ └─ adapter/

│ │ ├─ LocationAdapter.java

│ │ └─ LocationViewHolder.java

│ ├─ forecast/

│ │ ├─ ForecastFragment.java

│ │ ├─ ForecastViewModel.java

│ │ └─ adapter/

│ │ ├─ ForecastAdapter.java

│ │ └─ ForecastViewHolder.java

│ └─ sports/

│ ├─ SportsFragment.java

│ ├─ SportsViewModel.java

│ └─ adapter/

│ ├─ SportsAdapter.java

│ └─ SportsViewHolder.java

│

├─ domain/ // Lógica de negocio (modelos limpios + casos de uso)

│ ├─ model/

│ │ ├─ LocationItem.java

│ │ ├─ ForecastDayItem.java

│ │ └─ MatchItem.java

│ └─ usecase/

│ ├─ SearchLocationsUseCase.java

│ ├─ GetForecastUseCase.java

│ └─ GetFootballUseCase.java

│

├─ data/ // Acceso a datos

│ ├─ remote/

│ │ ├─ WeatherApi.java // Retrofit interface (o HttpURLConnection básico)

│ │ ├─ NetworkClient.java // Builder Retrofit/OkHttp

│ │ └─ dto/

│ │ ├─ LocationDto.java

│ │ ├─ ForecastDto.java

│ │ └─ SportsDto.java

│ ├─ repository/

│ │ └─ WeatherRepositoryImpl.java

│ └─ mapper/

│ ├─ LocationMapper.java

│ ├─ ForecastMapper.java

│ └─ SportsMapper.java

│

├─ core/ // Utilidades transversales

│ ├─ Result.java // Wrapper Success/Failure (genérico)

│ ├─ NetworkMonitor.java // ConnectivityManager.NetworkCallback → LiveData<Boolean>

│ ├─ Dialogs.java // Diálogo “Ir a Configuración” cuando no hay red

│ ├─ ExecutorsProvider.java // Executors para background (testable)

│ └─ constants/

│ ├─ ApiKeys.java

│ └─ Endpoints.java

│

└─ storage/ (opcional)

├─ cache/ // In-memory cache simple

└─ preferences/

└─ AppPreferences.java // SharedPreferences (última búsqueda, ciudad reciente)

Prompt 2

Genera código en Java para Android Studio. Proyecto com.example.lab3\_20211602\_iot.  
Quiero un NetworkMonitor que use ConnectivityManager.NetworkCallback para saber si hay Internet o no.  
Debe exponer un LiveData<Boolean> que pueda usarse en MainActivity con ViewBinding.  
También necesito un ConnectivityBanner (vista simple) que muestre un aviso cuando no hay red.  
Incluye permisos necesarios en el AndroidManifest.xml.

Prompt 3

Genera código en Java para Android Studio. Proyecto com.example.lab3\_20211602\_iot.  
Quiero un cliente de clima (WeatherRepository) que:

* Haga una petición GET con HttpURLConnection a una URL de ejemplo.
* Devuelva un objeto Forecast con ciudad, temperatura, descripción e icono.
* Guarde el resultado en SharedPreferences con un TTL de 10 minutos.
* Si el cache es válido lo devuelve, si no, hace la petición.  
  Incluye también MainActivity que use ViewBinding para mostrar el clima en pantalla.  
  Agrega permiso INTERNET en el AndroidManifest.xml.